Leveraging variant annotations for WGS data analysis

Deepti Jain & Ben Heavner

Data coordinating center for
Trans-Omics for Precision Medicine (TOPMed) program,
University of Washington

Deepti Jain (jaind@uw.edu) and Ben Heavner (bheavner@uw.edu)

Overview of variant annotation session

Section I : Thursday afternoon (instructional part)

- Why do we need annotations?
- What are variant annotations?
- Approaches for aggregating and filtering variants for rare variant association testing
- Generating variant grouping files for conducting rare-variant aggregate test
- WGSAparsr

Section II: Friday morning (hands–on part)

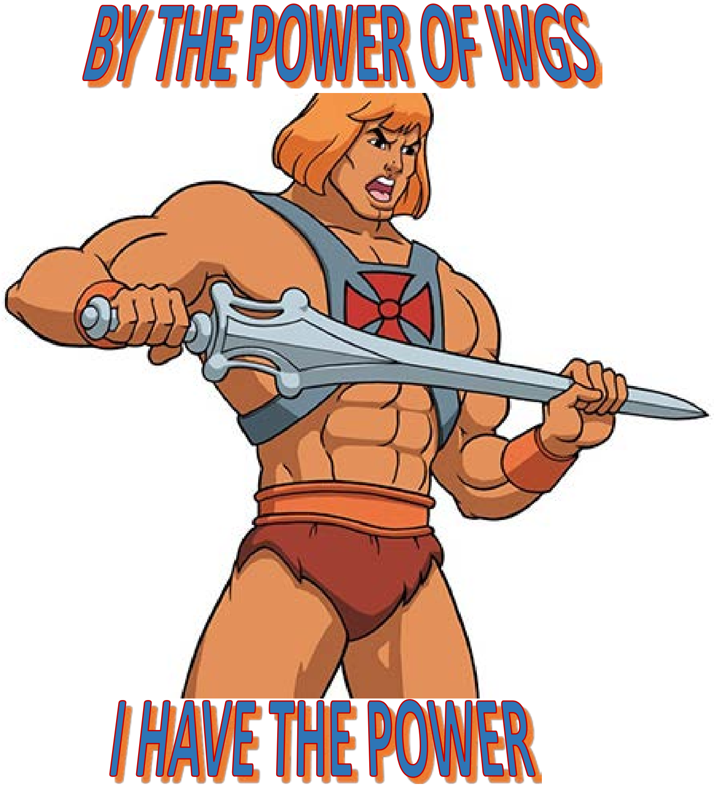
- Parsing WGSA files using WGSAparsr
- Generate variant grouping files
- Association testing in aggregation units using variant grouping files

Why do we need annotations?

1. Rare variant aggregate association tests
    - To define aggregation units
    - Filter aggregation units
    - Used as weights

2. Fine map novel and previously known significantly associated loci to identify likely causal variants

Why aggregate rare variants?

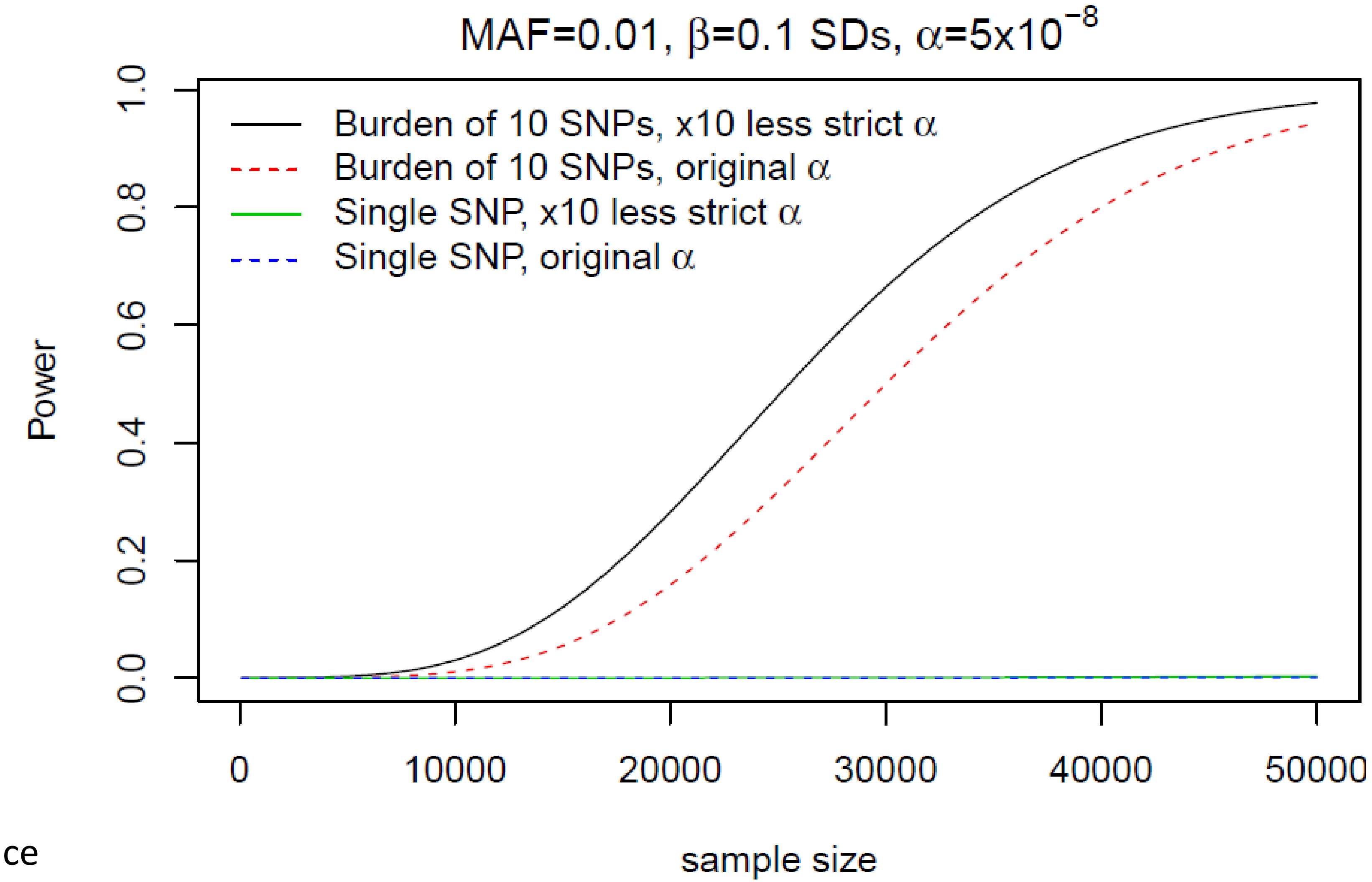


- Rare variant association tests lack power due to
  - an increased multiple testing burden
  - a decrease in statistical power owing to the rarity of individuals carrying these variant alleles

- To gain power, rare variants are generally combined within units of association which are referred as aggregation units

- Rare variants are aggregated typically in a biologically relevant region (example gene)

Aggregating variants boosts power



From Dr. Ken Rice

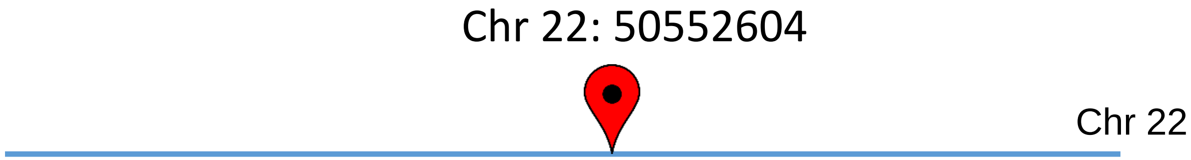
Section I : Outline

- Why do we need annotations?
- What are variant annotations?
- Approaches for generating aggregation units
- Generating variant grouping files for conducting rare-variant aggregate test
- WGSAparsr

What is annotation?

- Annotation is defined as a note of explanation or comment added to a text or diagram.
- Variant annotation is a note or comment about a specific variant
- Examples of variant annotation values include :
    - Gene a variant overlaps with
    - rs identifier of the variant
    - Conservation score (example GERP score) associated with a variant
    - Consequence associated with the variant (example non-synonymous)
    - Many others …..
- Annotations provide information about the variants which helps us to analyze and interpret them

Annotating variants = generating google maps for genome

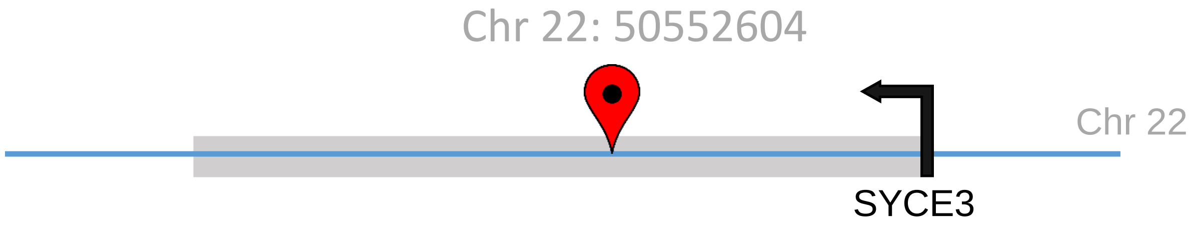


Name of the city provides some context about your location on Earth!

Chromosome and position provides some context about your location in the genome

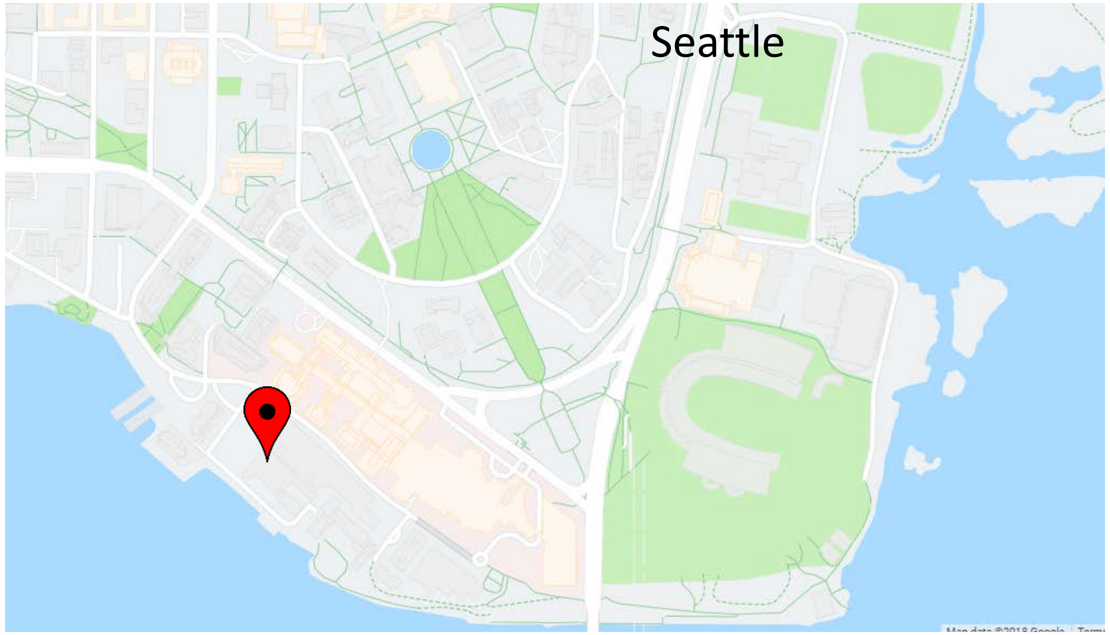
Annotating variants = generating google maps for genome

Building outlines overlaid indicate you are in a building

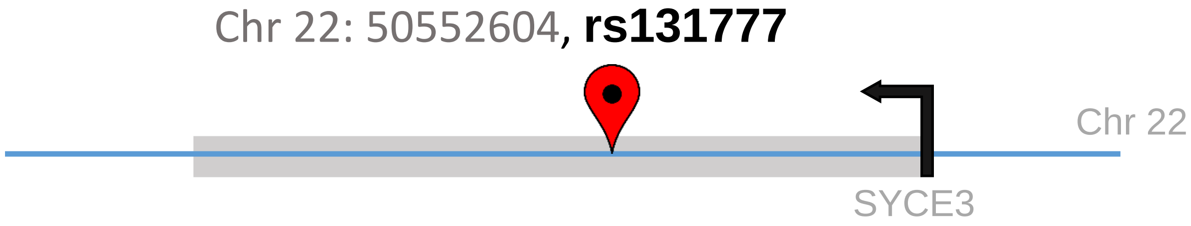


Gene name annotations identify that the variant overlaps with SYCE3 gene

Annotating variants = generating google maps for genome

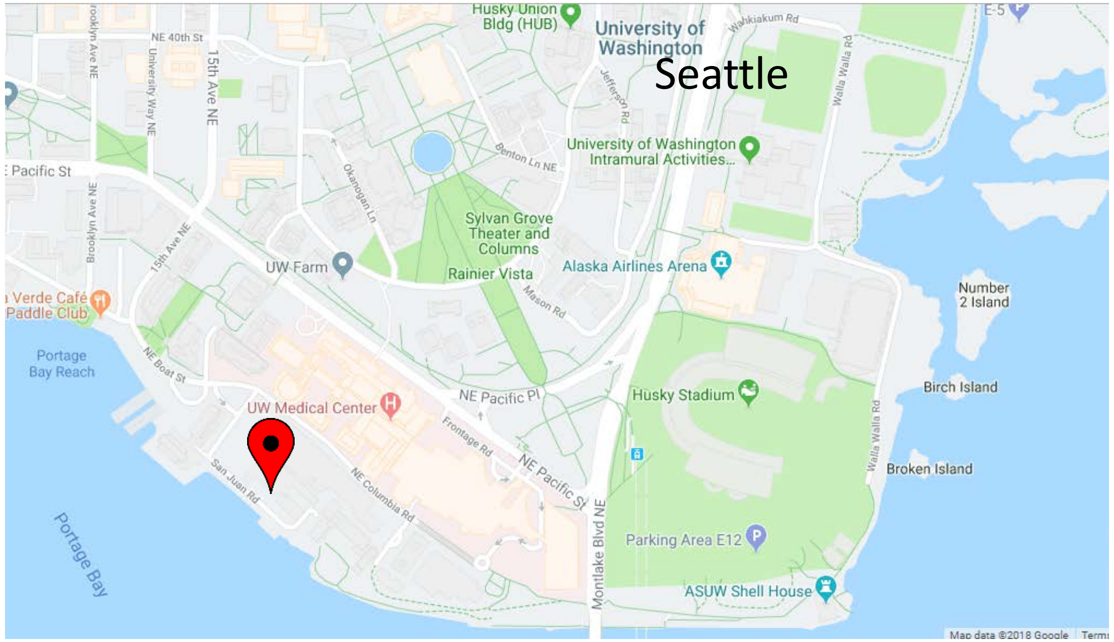


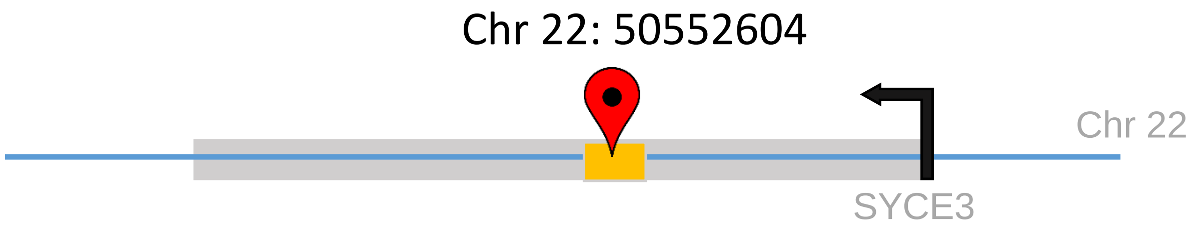
Roads overlaid show paths you can take to go from point A to B



rs identifier and GWAS catalogue annotations help you identify that this variant is previously associated with red cell trait “Mean corpuscular volume”

Annotating variants = generating google maps for genome





Road and building names overlaid

- You can take a walk to UW farm
- You can get lunch at Agua verde
- You can go the Husky stadium

Regulatory annotations help you identify that the variant overlaps with a regulatory element. The overlapping regulatory element is active in[*] :

- Red cell cells
- Platelets
- Not in brain cells and bladder cells

[*] hypothetical example

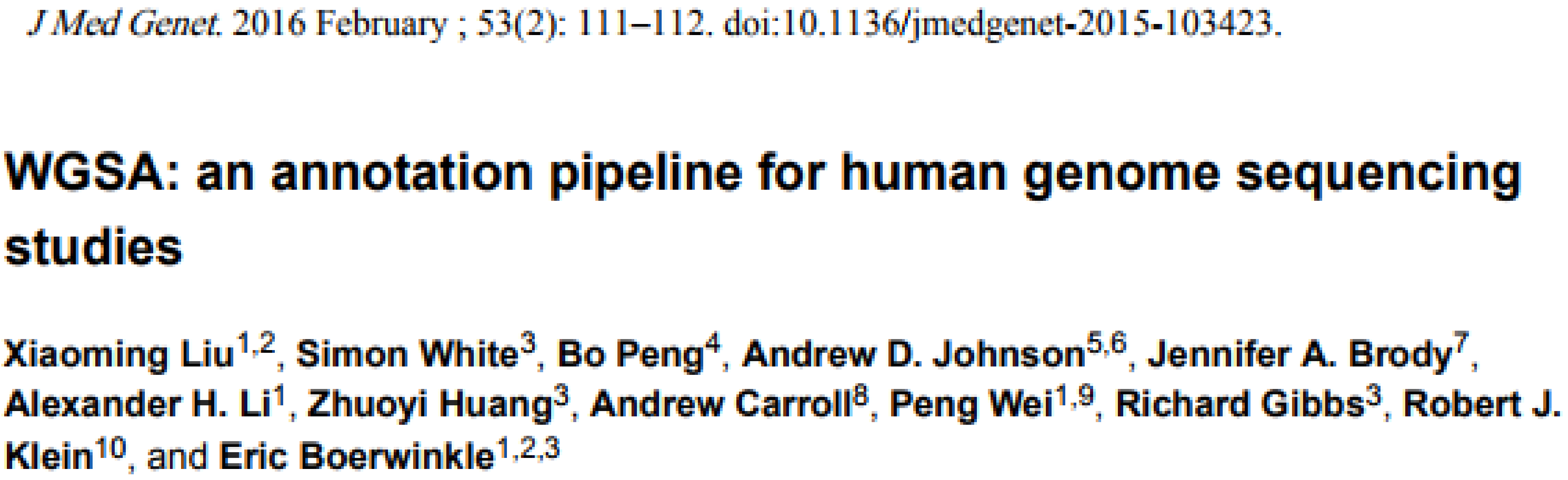
What is the source of annotations?

- Lots of resources!

Annotation can be generated by any lab or consortium

- NCBI
- Ensemble
- UCSC
- ENCylopedia Of DNA Elements (ENCODE)
- Roadmap Epigenomics Consortium
- FANTOM5
- dbSNP
- …

WGSA



J Med Genet. 2016 February ; 53(2): 111–112. doi:10.1136/jmedgenet-2015-103423.

WGSA: an annotation pipeline for human genome sequencing studies

Xiaoming Liu[1,2], Simon White[3], Bo Peng[4], Andrew D. Johnson[5,6], Jennifer A. Brody[7], Alexander H. Li[1], Zhuoyi Huang[3], Andrew Carroll[8], Peng Wei[1,9], Richard Gibbs[3], Robert J. Klein[10], and Eric Boerwinkle[1,2,3]

- [Website: https://sites.google.com/site/jpopgen/wgsa/](https://sites.google.com/site/jpopgen/wgsa/)
- WGSA is provided both as
  - an Amazon Machine Image (AMI) ready to run out-of-the-box and
  - a downloadable version
- Licenses are required for non-academic usage for some of the resources

WGSA has over 1,500 annotations

- Gene based location and consequence
  - Softwares : SnpEff, ANNOVAR, VEP
  - Gene models: Ensembl ,RefSeq ,UCSC
- Transcript-specific annotation (transcript name, consequence etc.)
- Loss-of-function annotations (eg: LOFTEE)
- Deleteriousness predictions(CADD, MetaSVM, ssSNV etc)
- Allele frequencies (1000G, UK10K, EXAC, gnomAD etc)
- Regulatory annotations (ENCODE, Roadmap, FANTOM5)
- Conservation scores (GERP etc)
- Mappability scores
- rsIDs
- Many more ….

Section I : Outline

- Why do we need annotations?
- What are variant annotations?
- Approaches for generating aggregation units
- Generating variant grouping files for conducting rare-variant aggregate test
- WGSAparsr

Why do we need annotations?

1. Rare variant aggregate association tests

2. Fine map novel and previously known significantly associated loci to identify likely causal variants

Two steps involved in generating aggregated variant list for association testing

STEP1: Define aggregation units

- which genomic regions will be included in each unit

STEP2: Decide on filtering criteria

- which variants will be filtered within each unit

Goal is to create list of variants in each aggregation unit which can be used in multiple variants association tests (example Burden and SKAT tests)

STEP1: Define aggregation units

Gene is one of the fundamental units of biology and gene-based aggregation units are frequently used in rare variant association testing so we will go over these in detail

Gene based aggregation units

- Gene and/or gene related elements are the unit of aggregation

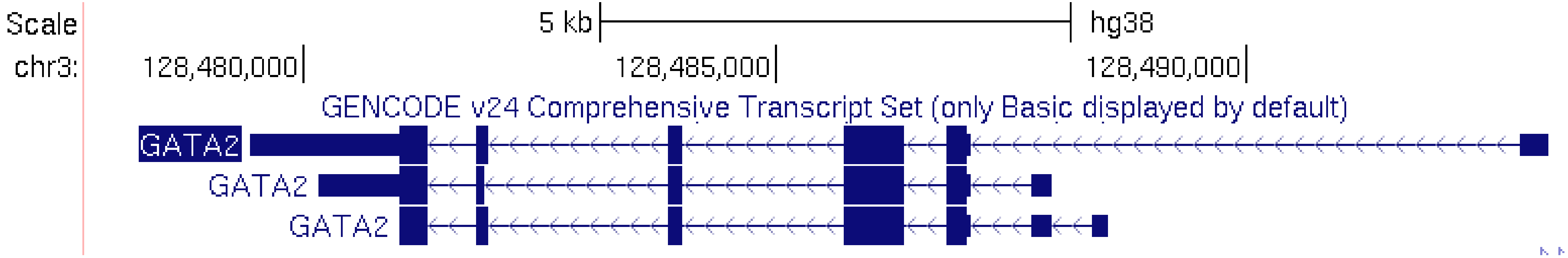
- Multiple gene models available
  - GENCODE/Ensembl, RefSeq and UCSC

- Multiple releases of a given gene model are available for same genome build
  - GENCODE v24,v26 etc. on same genome build

- If possible try and use gene specific annotations from the same gene model and version across the analyses

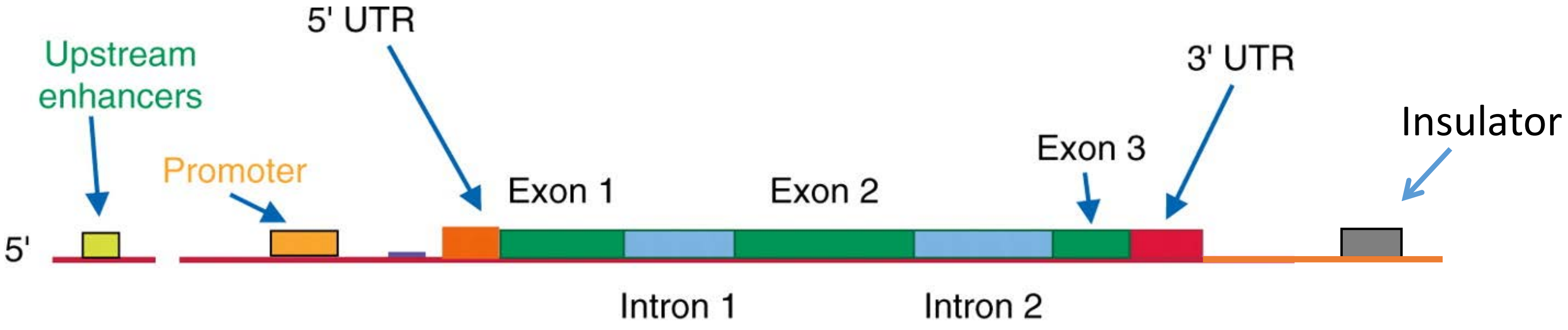
Gene as the aggregation unit

- Gene is the contiguous genomic region spanning all the transcripts of a gene


Scale
5 kb
hg38
chr3:
128,480,000
128,485,000
128,490,000
GENCODE v24 Comprehensive Transcript Set (only Basic displayed by default)
GATA2
GATA2
GATA2

Biotype	Unique Ensembl identifier	Genomic coordinates
Gene	ENSG00000179348	Chromosome 3: 128,479,427-128,493,185
Transcript	ENST00000341105	Chromosome 3: 128,479,427-128,493,185
Transcript	ENST0000043026	Chromosome 3: 128,480,146-128,487,916
Transcript	ENST00000487848	Chromosome 3: 128,481,019-128,488,530

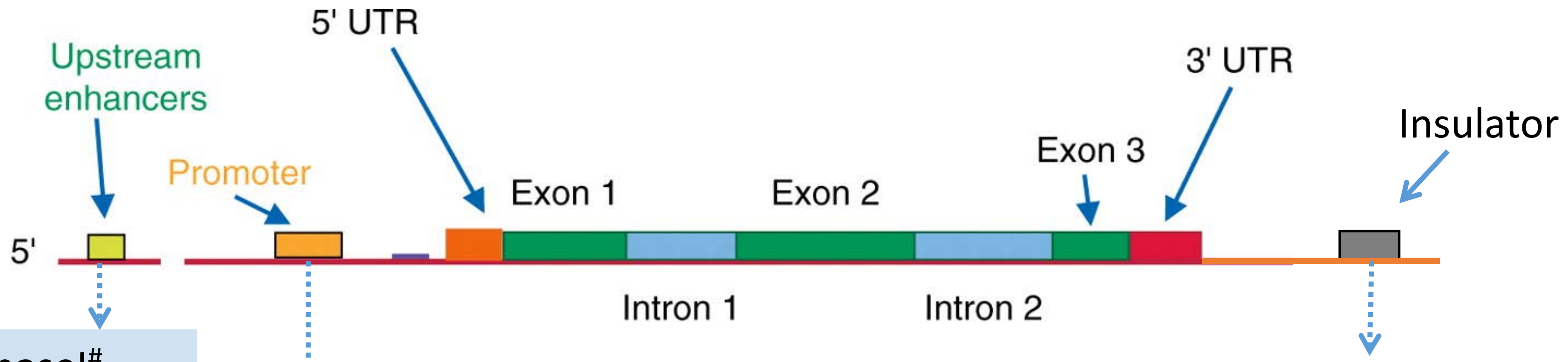
Gene is regulated by non-coding regulatory elements



Functional gene unit = transcript + its regulatory elements

Modified from R. Searle and P. M. Hopkins, 2009

Biochemical signatures typically associated with non-coding functional elements



DnaseI#
H3k4me1 mark
H3k27ac mark
TF* binding

DNaseI#
H3k4me3 mark
H3k27ac mark
Polymerase II
TF* binding

DNaseI#
CTCF binding

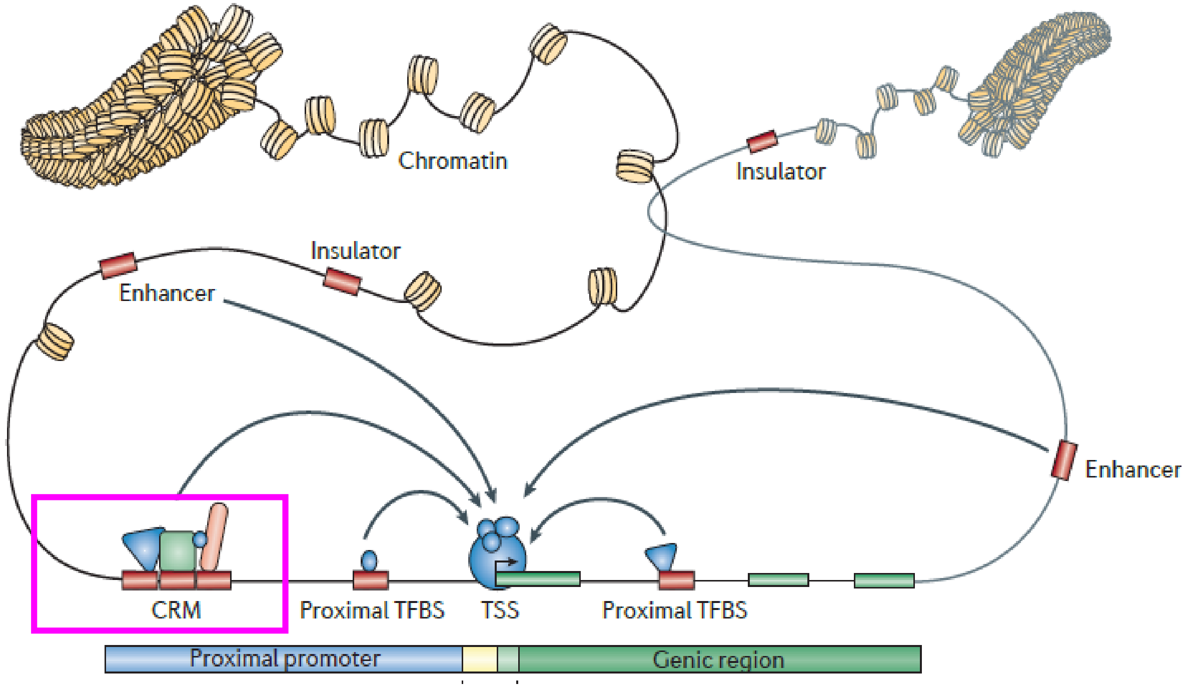
Enhancer : Interacts with promoter can be involved in repression or induction of a gene
Promoter : Genomic element where the transcription machinery assembles
UTR **:** Untranslated region
EXON : Coding part of a transcript (mRNA)
INTRON : Non-Coding part of a transcript (mRNA)
Insulator : Barriers that protect genes from influence of outside enhancers or inactivating chromatin structures

TF : transcription factor,
# DNaseI Hypersensitivity, which is an indicator of chromatin accessibility

NOTE: These biochemical marks are tissue-specific . Additionally, these may also show temporal and treatment specific variations within a cell type

Modified from R. Searle and P. M. Hopkins, 2009

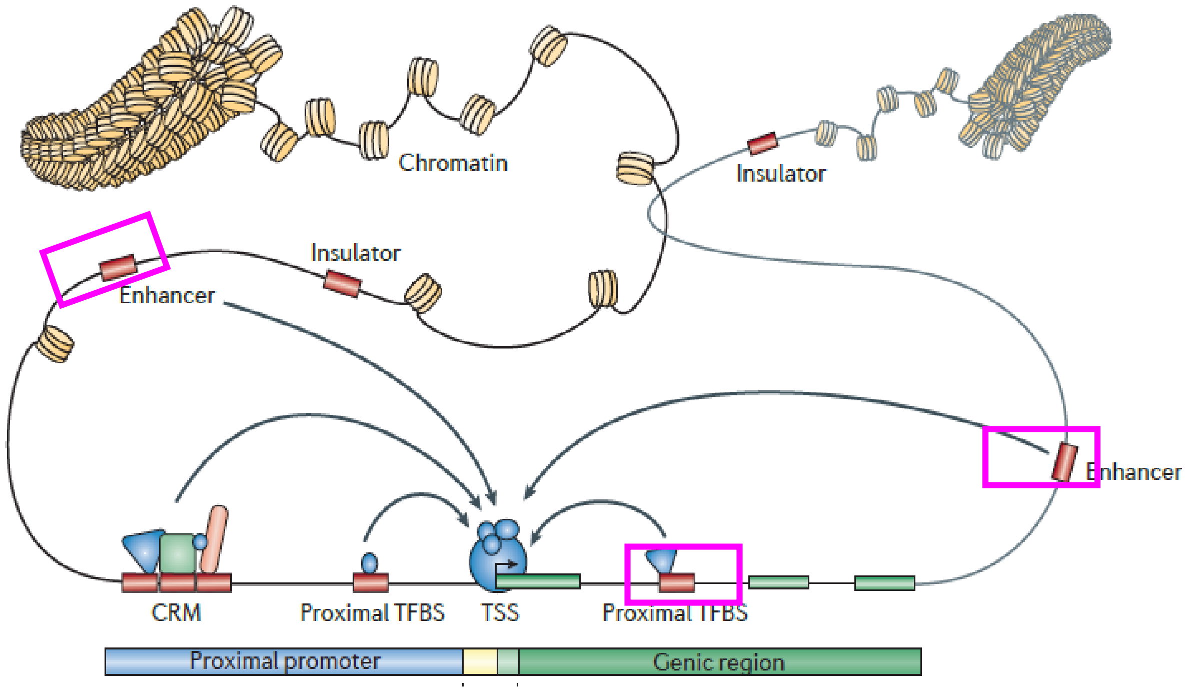
Promoters



- Some distance upstream from TSS (typically 5Kb)
- 5Kb upstream overlapping with H3K4me3 and or H3K27ac mark
- 5Kb upstream overlaps with DNaseI hypersensitive regions
- 5Kb upstream that overlaps with CAGE peaks[1]

[1] Morrison AC, Huang Z, Yu B, et al. Practical Approaches for Whole-Genome Sequence Analysis of Heart- and Blood-Related Traits. Am J Hum Genet. 2017;100(2):205-215.

Enhancers



- Flanking regions overlapping with H4K4me1 and or H3K27ac
- Flanking regions overlapping with DNaseI hypersensitive regions
- Enhancer-gene link predictions[1,2, 3]
- Chromosome conformation capture (3C,4C,Hi-C etc.)

[1] *Thurman RE et.al Nature. 2012 Sep 6; 489(7414):75-82.*

[2] *Forrest AR, Kawaji H, Rehli M, et al. A promoter-level mammalian expression atlas. Nature. 2014;507(7493):462-70.*

[3] *Fishilevich et.al. GeneHancer: genome-wide integration of enhancers and target genes in GeneCards, Database, 2017*

Example gene-based aggregation units

- Gene
- Gene + flanking regions
- Gene + enhancer(s) + promoter
- UTR’s+ enhancer(s) + promoter
- Promoter of a gene
- First intron of a transcript

Other approaches for aggregating variants

Aggregation units are defined based on genomic positions and they can be :

1. Contiguous units of aggregation
    - Moving window
    - Gene
    - Transcript
    - Exons
    - introns
    - Regulatory regions
        - Promoters
        - Enhancers
        - DNAse hypersensitive site (DNAse sites)
        - Transcription factor binding sites (TFBSs)
        - ChromHMM states
    - Topologically associated domains (TAD's)

2. Non-contiguous units of aggregation
    - Gene/Transcript + its associated regulatory regions
    - Domains of interacting proteins
    - Genes in a pathway

Any other biologically motivated unit of your choice …

STEP2: Filtering aggregation units

Filtering aggregation units

- Variants within aggregated regions are filtered
  - to Increase the proportion of likely causal variants
  - Good filtering strategy increases the signal to noise ratio & increases power to detect an association
- Typically, one or a combination of annotations are used for filtering
  - Example: Within a gene keep variants that
    - a) Are frameshift mutations or
    - b) Overlap Genomic Evolutionary Rate Profiling (GERP) score > 0 or
    - c) Overlap with transcript factor binding sites in blood cells
- Various permutation and combinations of filtering are possible
- The choice of filtering will depend on the goal of your analysis

Scenario 1: simple filtering

- Genic unit

  Transcript range + 20 kb flanking region upstream and downstream

- Filters:

  FATHMM-MKL>=0.5 and MAF<=1%

- Functional Analysis Through Hidden Markov Models -Multiple Kernel Learning (FATHMM-MKL)
  - FATHMM-MKL generates scores predicting functional consequences of both coding and non-coding sequence. FATHMM-MKL is a machine learning approach that integrates functional annotations from ENCODE with nucleotide based sequence conservation measures variants.
  - FATHMM-XF (FATHMM with eXtended Features) represents a substantial improvement over FATHMM-MKL

Dong C, et.al. Hum Mol Genet, 2015
Shihab HA, et.al, Bioinformatics, 2015

Scenario 2: Filtering using multi-tissue regulatory regions

- Genic unit

  Gene + 20 kb flanking region upstream and downstream

- Filters:

  A. Flanking region
     - Overlaps with “Ensembl_Regulatory_Build_Overviews”

  A. Gene region
     - Overlaps with “Ensembl_Regulatory_Build_Overviews” OR
     - Overlaps with LOF variants

- Ensembl_Regulatory_Build_Overviews
  - genome segment prediction based on 17 cell types from ENCODE and Roadmap.
  - ctcf - CTCF binding sites,
  - distal - Predicted enhancers
  - open - Unannotated open chromatin regions
  - proximal - Predicted promoter flanking regions
  - tfbs - Unannotated transcription factor binding sites
  - tss - Predicted promoters

- ENCODE_Dnase_cells: number of cell lines supporting a DNase I hypersensitive site

Scenario 3: Using tissue specific regulatory regions

- Genic unit

  Gene + 20 kb flanking region upstream and downstream

- Filters:

  A. Flanking region

  - Overlaps with either H3K4me3 or H3K4me1 enriched regions) & DNaseI hypersensitivity sites in k562 cells

  B. Gene region

  - overlap (either H3K4me3 or H3K4me1 enriched regions) & DNaseI hypersensitivity sites in k562 cells OR
  - Overlaps with LOF variants

Using quantitative annotation scores for filtering

- Aggregation units can be further filtered using one or several quantitative annotation scores
  - Conservation scores (GERP,phyloP)
  - Prediction scores for deleteriousness (CADD, fathmm_MKL)
  - Consequence terms (missense, frameshift, etc.)
  - Regulatory annotation (DNAse sites, TFBSs, etc.)
  - Score predicting impact on variation on protein function (SIFT, polyPhen)
  - Many others ..
- Quantitative annotation scores can also be used as weights in the association model to avoid the stringent score based filtering[1]

[1] Morrison AC, Huang Z, Yu B, et al. Practical Approaches for Whole-Genome Sequence Analysis of Heart- and Blood-Related Traits. Am J Hum Genet. 2017;100(2):205-215.

Section I : Outline

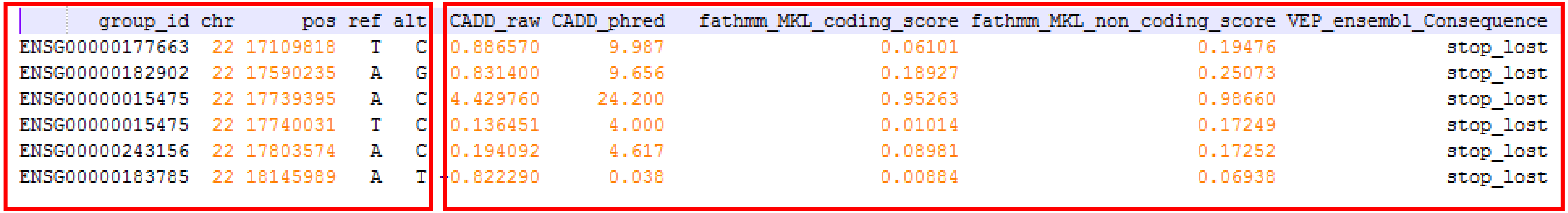
- Why do we need annotations?
- What are variant annotations?
- Approaches for generating aggregation units
- Generating variant grouping files for conducting rare-variant aggregate test
- WGSAparsr

Filtered list of variants grouped by aggregation unit are stored in variant grouping files

- Variant-level grouping file example from the TOPMed DCC pipeline
- Variants aggregated over gene and filtered to keep Loss of Function variants

Required fields

Additional annotation fields



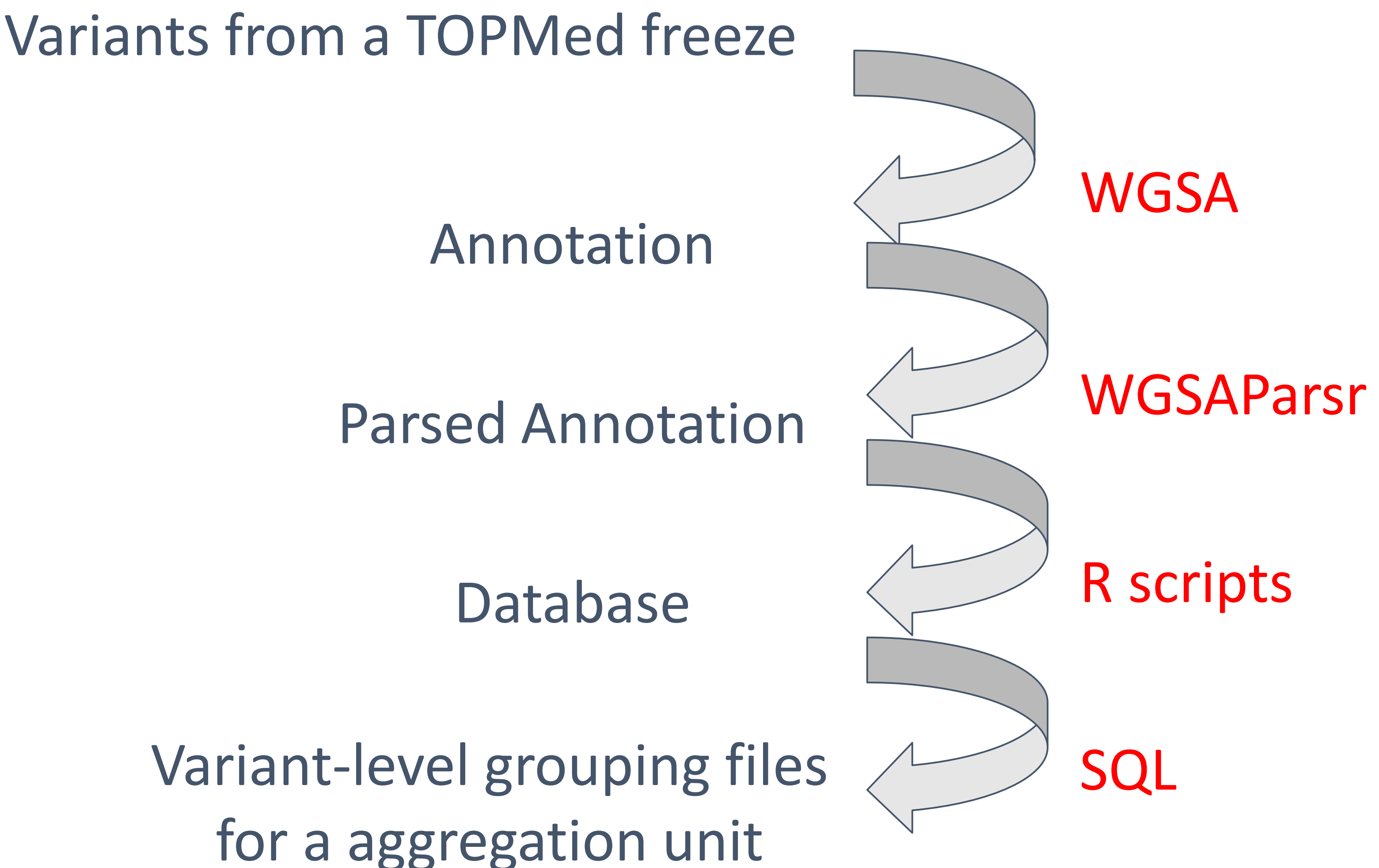
group_id	chr	pos	ref	alt	CADD_raw	CADD_phred	fathmm_MKL_coding_score	fathmm_MKL_non_coding_score	VEP_ensembl_Consequence
ENSG00000177663	22	17109818	T	C	0.886570	9.987	0.06101	0.19476	stop_lost
ENSG00000182902	22	17590235	A	G	0.831400	9.656	0.18927	0.25073	stop_lost
ENSG00000015475	22	17739395	A	C	4.429760	24.200	0.95263	0.98660	stop_lost
ENSG00000015475	22	17740031	T	C	0.136451	4.000	0.01014	0.17249	stop_lost
ENSG00000243156	22	17803574	A	C	0.194092	4.617	0.08981	0.17252	stop_lost
ENSG00000183785	22	18145989	A	T	0.822290	0.038	0.00884	0.06938	stop_lost

Aggregation unit identifier :
For gene based units it's
the ENSG gene identifier

Variant information

Workflow for generating variant grouping files in TOPMed

- In the TOPMed DCC pipeline filtered aggregation units are passed to the pipeline as variant grouping files



Variants from a TOPMed freeze

WGSA

Annotation

WGSAParsr

Parsed Annotation

R scripts

Database

SQL

Variant-level grouping files for a aggregation unit

Working with variant-level grouping files

TOPMed DCC analysis Pipeline:

- The variant-level grouping file are .Rdata files which can be used with the DCC analysis pipeline[1] directly
- File name is passed as a value to the config parameter “variant_group_file”

GENESIS :

- variant-level grouping files can be processed using the function TopmedPipeline[2]::aggregateGRangesList to produce a format suitable for GENESIS[3] function assocTestAggregate
- See scripts below for an example of implementation
  - https://github.com/UW-GAC/analysis_pipeline/blob/devel/R/aggregate_list.R
  - https://github.com/UW-GAC/analysis_pipeline/blob/devel/R/assoc_aggregate.R

[1] https://github.com/UW-GAC/analysis_pipeline,
[2] https://github.com/UW-GAC/analysis_pipeline/tree/devel/TopmedPipeline,
[3] https://github.com/smgogarten/GENESIS

Why do we need annotations?

1. Rare variant aggregate association tests
   - To define aggregation units
   - Filter aggregation units
   - Used as weights

2. Fine map novel and previously known significantly associated loci to identify likely causal variants
   - Annotation can be used in fine mapping software's like PAINTOR* for identifying likely causal variants
   - Annotations can be explored manually to prioritizing variants for experimental follow-up

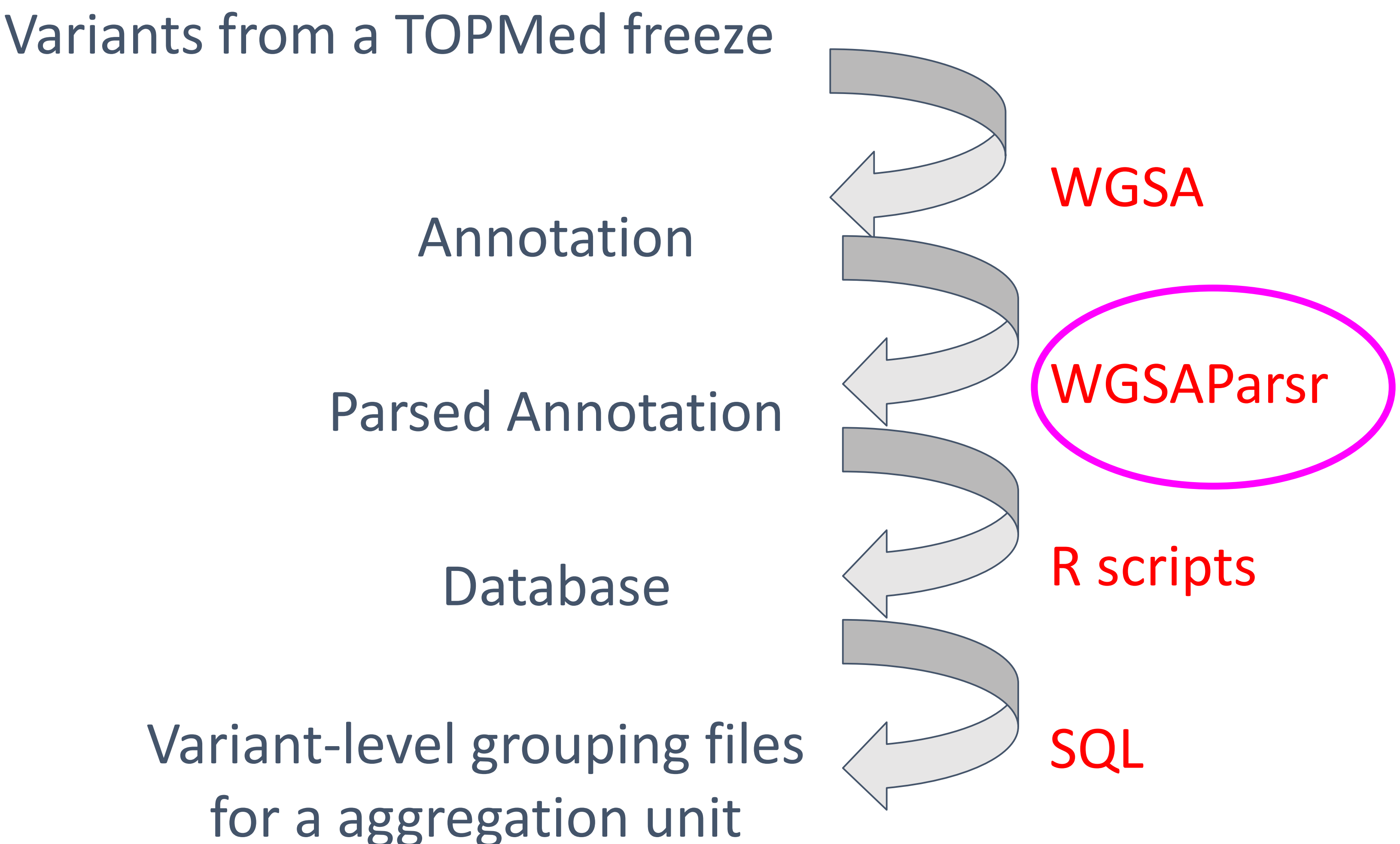
* Kichaev G, et.al, Bioinfromatics, 2017

Section I : Outline

- Why do we need annotations?
- What are variant annotations?
- Approaches for generating aggregation units
- Generating variant grouping files for conducting rare-variant aggregate test
- WGSAparsr

Workflow for generating variant grouping files in TOPMed

- In the TOPMed DCC pipeline filtered aggregation units are passed to the pipeline as variant grouping files



Variants from a TOPMed freeze

WGSA

Annotation

WGSAParsr

Parsed Annotation

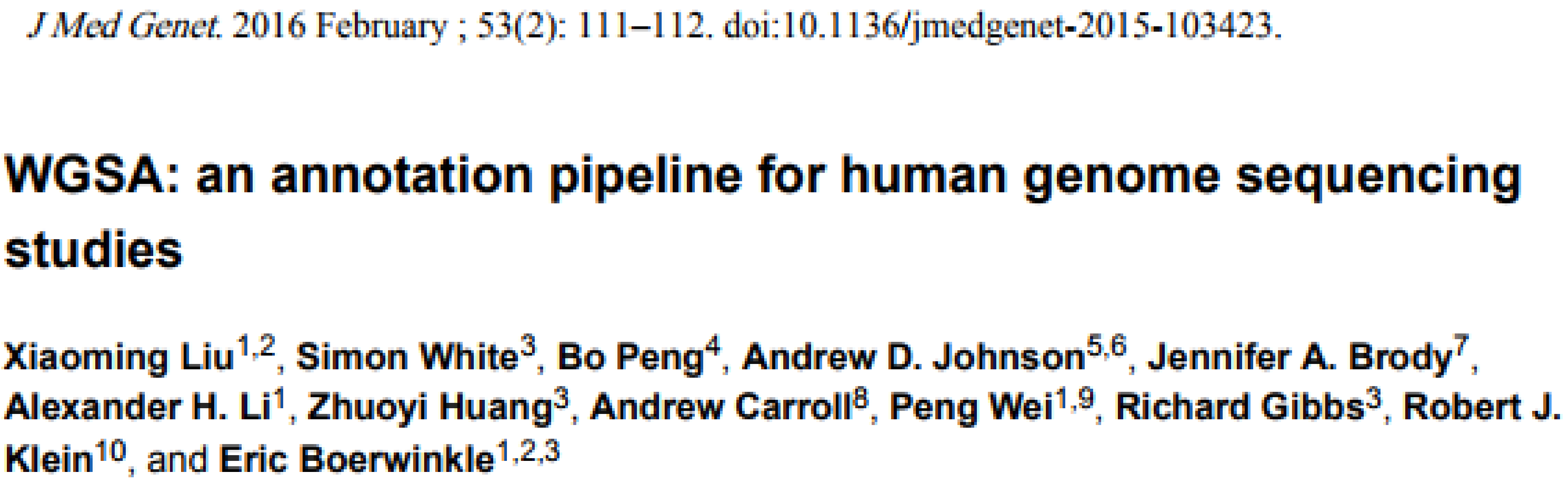
R scripts

Database

SQL

Variant-level grouping files
for a aggregation unit

WGSA



J Med Genet. 2016 February ; 53(2): 111–112. doi:10.1136/jmedgenet-2015-103423.

WGSA: an annotation pipeline for human genome sequencing studies

Xiaoming Liu[1,2], Simon White[3], Bo Peng[4], Andrew D. Johnson[5,6], Jennifer A. Brody[7], Alexander H. Li[1], Zhuoyi Huang[3], Andrew Carroll[8], Peng Wei[1,9], Richard Gibbs[3], Robert J. Klein[10], and Eric Boerwinkle[1,2,3]

- Built and maintained by Xiaoming Liu
- https://sites.google.com/site/jpopgen/wgsa
- The latest TOPMed freeze has ~ 800 million variants
- The WGSA files are 432G

WGSA annotations are complex

- Several annotations are compound entries
  - Example : **VEP_ensembl_Transcript_ID**
    - ENST00000456328|ENST00000488147|ENST00000438504|ENST00000515242|ENST00000541675|ENST00000423562|ENST00000450305|ENST00000538476|ENST00000518655

- Some annotations for indels have multiple values
  - Example : GERP score
    - indel: chr1:12729:GAGAGT: G
    - GERP score : .{1}-0.824{1}-0.943{1}0.472{2}

- We often only work with only subset of the annotations

Gene-based annotations in WGSA output are at transcript level

chr:10273 T>C

VEP_ensembl_Transcript_ID

ENST00000456328|ENST00000488147|ENST00000438504|ENST00000515242|ENST00000541675|ENST00000423562|ENST00000450305|ENST00000538476|ENST00000518655

VEP_ensembl_Consequence

upstream_gene_variant|downstream_gene_variant|downstream_gene_variant|upstream_gene_variant|downstream_gene_variant|downstream_gene_variant|upstream_gene_variant|downstream_gene_variant|*splice_region_varian*t

VEP_ensembl_Gene_Name

DDX11L1|WASH7P|WASH7P|DDX11L1|WASH7P|WASH7P|DDX11L1|WASH7P|DDX11L1

VEP_ensembl_Gene_ID

ENSG00000223972|ENSG00000227232|ENSG00000227232|ENSG00000223972|ENSG00000227232|ENSG00000227232|ENSG00000223972|ENSG00000227232|ENSG00000223972

Ensembl_Regulatory_Build_Overviews

ctcf

VEP_ensembl_LoF

.|.|.|.|.|.|.|.|HC

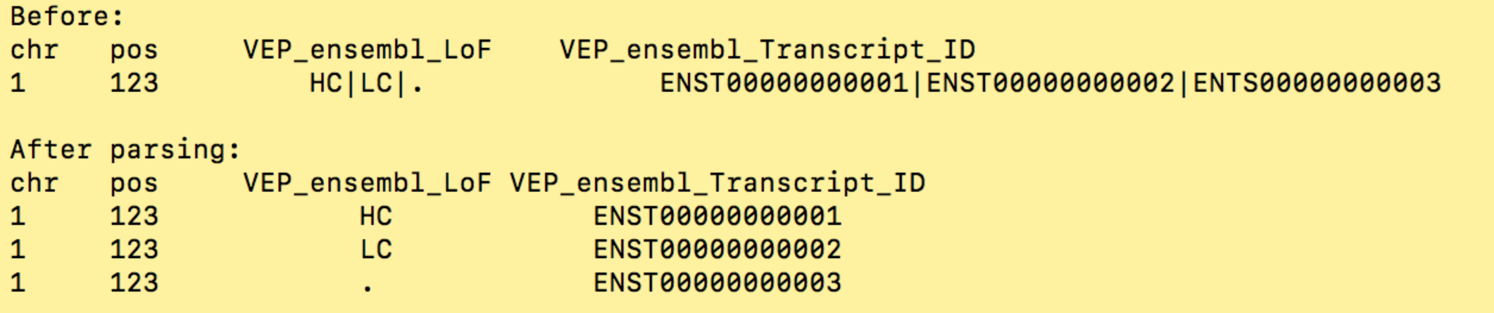
WGSAparsr

- We need to simplify the WGSA output so that we can easily parse these complex annotation files

- **WGSAparsr**: an R package built and maintained by the Ben Heavner
  - https://github.com/UW-GAC/wgsaparsr

WGSAParsr operations

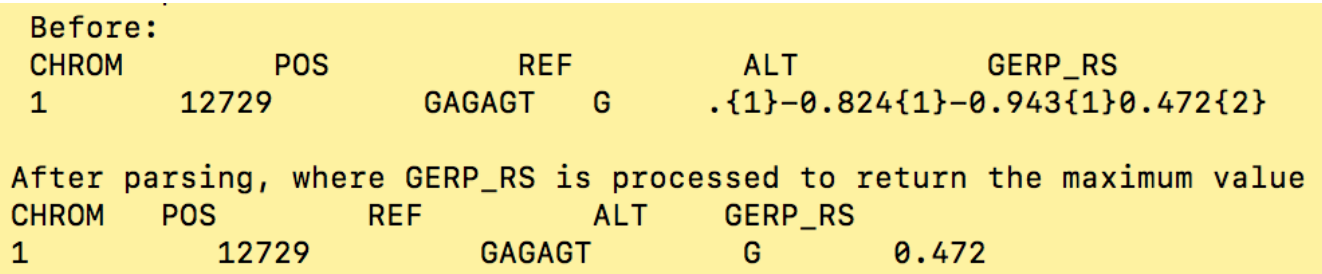
1. Selecting fields
2. Renaming fields
3. Simplifying fields
    a. Pivoting



```
Before:
chr   pos     VEP_ensembl_LoF    VEP_ensembl_Transcript_ID
1     123         HC|LC|.              ENST00000000001|ENST00000000002|ENTS00000000003

After parsing:
chr   pos     VEP_ensembl_LoF VEP_ensembl_Transcript_ID
1     123             HC            ENST00000000001
1     123             LC            ENST00000000002
1     123             .             ENST00000000003
```

    a. Selecting values



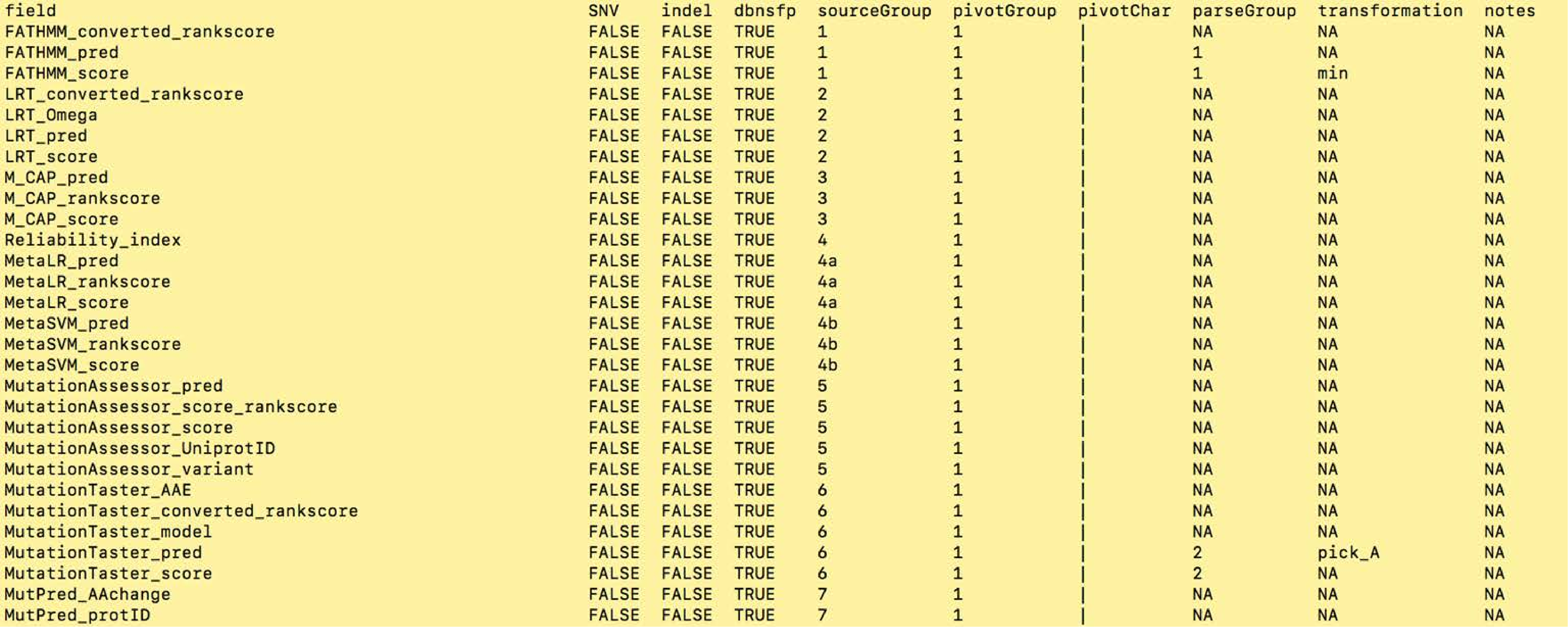
```
 Before:
 CHROM         POS           REF          ALT          GERP_RS
 1        12729         GAGAGT    G     .{1}-0.824{1}-0.943{1}0.472{2}

After parsing, where GERP_RS is processed to return the maximum value
CHROM    POS          REF          ALT     GERP_RS
1           12729          GAGAGT         G        0.472
```

Using WGSAParsr

```
# parse snv and dbnsfp:
parse_to_file(source_file = snv_source_file,
             destination = snv_destination,
             dbnsfp_destination = dbnsfp_destination,
             config = config,
             freeze = 5,
             chunk_size = 1000,
             verbose = TRUE)
```

Using WGSAParsr - Configuration file



```
field                                                   SNV    indel  dbnsfp  sourceGroup  pivotGroup  pivotChar  parseGroup  transformation  notes
FATHMM_converted_rankscore                              FALSE  FALSE  TRUE    1            1           |          NA          NA              NA
FATHMM_pred                                             FALSE  FALSE  TRUE    1            1           |          1           NA              NA
FATHMM_score                                            FALSE  FALSE  TRUE    1            1           |          1           min             NA
LRT_converted_rankscore                                 FALSE  FALSE  TRUE    2            1           |          NA          NA              NA
LRT_Omega                                               FALSE  FALSE  TRUE    2            1           |          NA          NA              NA
LRT_pred                                                FALSE  FALSE  TRUE    2            1           |          NA          NA              NA
LRT_score                                               FALSE  FALSE  TRUE    2            1           |          NA          NA              NA
M_CAP_pred                                              FALSE  FALSE  TRUE    3            1           |          NA          NA              NA
M_CAP_rankscore                                         FALSE  FALSE  TRUE    3            1           |          NA          NA              NA
M_CAP_score                                             FALSE  FALSE  TRUE    3            1           |          NA          NA              NA
Reliability_index                                       FALSE  FALSE  TRUE    4            1           |          NA          NA              NA
MetaLR_pred                                             FALSE  FALSE  TRUE    4a           1           |          NA          NA              NA
MetaLR_rankscore                                        FALSE  FALSE  TRUE    4a           1           |          NA          NA              NA
MetaLR_score                                            FALSE  FALSE  TRUE    4a           1           |          NA          NA              NA
MetaSVM_pred                                            FALSE  FALSE  TRUE    4b           1           |          NA          NA              NA
MetaSVM_rankscore                                       FALSE  FALSE  TRUE    4b           1           |          NA          NA              NA
MetaSVM_score                                           FALSE  FALSE  TRUE    4b           1           |          NA          NA              NA
MutationAssessor_pred                                   FALSE  FALSE  TRUE    5            1           |          NA          NA              NA
MutationAssessor_score_rankscore                        FALSE  FALSE  TRUE    5            1           |          NA          NA              NA
MutationAssessor_score                                  FALSE  FALSE  TRUE    5            1           |          NA          NA              NA
MutationAssessor_UniprotID                              FALSE  FALSE  TRUE    5            1           |          NA          NA              NA
MutationAssessor_variant                                FALSE  FALSE  TRUE    5            1           |          NA          NA              NA
MutationTaster_AAE                                      FALSE  FALSE  TRUE    6            1           |          NA          NA              NA
MutationTaster_converted_rankscore                      FALSE  FALSE  TRUE    6            1           |          NA          NA              NA
MutationTaster_model                                    FALSE  FALSE  TRUE    6            1           |          NA          NA              NA
MutationTaster_pred                                     FALSE  FALSE  TRUE    6            1           |          2           pick_A          NA
MutationTaster_score                                    FALSE  FALSE  TRUE    6            1           |          2           NA              NA
MutPred_AAchange                                        FALSE  FALSE  TRUE    7            1           |          NA          NA              NA
MutPred_protID                                          FALSE  FALSE  TRUE    7            1           |          NA          NA              NA
```

Documented in ?wgsaparsr::load_config()

Overview of variant annotation session

Section I : Thursday afternoon (instructional part)

- Why do we need annotations?
- What are variant annotations?
- Approaches for aggregating and filtering variants for rare variant association testing
- WGSAparsr

Section II: Friday morning (hands–on part)

- Parsing WGSA files using WGSAparsr
- Generate variant grouping files
- Association testing in aggregation units using variant grouping files

Recap

- We need annotations to increase power in rare variant association testing
- Approaches for defining aggregation units
- Approaches for filtering aggregation units
- Overview of how WGSAparsr works

Exercises for the hands-on Session

- Parsing WGSA files using WGSAparsr
- Generate variant grouping files
- Association testing in aggregation units using variant grouping files

Key libraries and functions

Parse the WGSA annotation file	
library (wgsaparsr)	Package for working with WGSA output files
get_fields()	List the annotation fields available in a WGSA output file
parse_to_file()	Parses WGSA output files using selection and transformation defined in configuration file
load_config()	Load a configuration file to an R data frame
Aggregate variants by genic units and create input file for association testing	
- R code for the workshop - DCC uses a MySQL server for creating and filtering variant list in aggregation units using WGSA annotations	